UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/713,872	11/14/2003	Matthias Eberhard Sohn	11884/405801	1867

25693 7590 01/28/2008

KENYON & KENYON LLP
RIVERPARK TOWERS, SUITE 600
333 W. SAN CARLOS ST.
SAN JOSE, CA 95110

EXAMINER	
VU, TUAN A	
ART UNIT	**PAPER NUMBER**
2193	

MAIL DATE	DELIVERY MODE
01/28/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/713,872	SOHN ET AL.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *16 November 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,2,4,5,10,12,15 and 17-24* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-2, 4-5, 10, 12, 15, 17-24* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/ are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 11/16/07.

As indicated in Applicant's response, claims 17-21 have been amended. Claims 2, 4-5, 10, 12, 15, 17-24 are pending in the office action.

Claim Rejections - 35 USC § 112

2. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

3. Claim 2, 4-5, 10, 12, 15, 17-24 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 17 recites

(i) "application framework in the second layer" (cl. 17, li. 5, li. 16);

(ii) 'application framework metadata ... occupying a first layer' (cl. 17, li 9, 24);

(iii) 'a repository framework model in the second layer' (cl. 17, li. 13-14, li. 27);

(iv) 'application object repository framework in the first layer' (cl. 7, li. 12-13).

It is observed that there is a large **inconsistency** as to what entity pertains to a *first* layer as opposed to being pertinent to a *second* layer: e.g. is *application framework metadata* or *application object repository framework* or both occupying the first layer? whether *application framework* or *repository framework model* or both represent the second layer? is there any translation in format (e.g. based on *defining*) so that the one entity moves to another layer? As the claim language is compounded with repetitious instances of framework-related entity names, as a whole, it is not reasonably understood inasmuch as to establishing a clear scenario as to how

these so-named entities are what they call for, let alone a clear sequence of transformations including interaction between these layer's entities. That is, the above framework-related nomenclature, inter alia, is not teaching how one given layer's entity is changed to another, thereby is moved from one layer to a next, rendering the named entities as identified above per layer largely unsubstantiated and indefinite. This confusion is not justifiably resolved by reading the Specifications. The *Specifications* describes M-level format of metadata, and layers (e.g. Figure 9, about template workbench in a Rose meta-model, validating templates and generating object repository of source file or scripts to yield a object repository runtime) but nowhere is showing a specific 'layer' in correlation with the entities **exactly as** phrased in the claim, let alone the confusion created by the M-level meta language (e.g. Figure 8) paradigm which also fails to depict **layer-pertinent** actions (e.g. *defining*, *generating*, *modeling* in relation to entities (i)→(iv) as claimed.

The Specifications do not provide a deliberate and clear description as to how these layers are defined with proper association thereof with each of the recited entities (i) to (iv) in order to enable one of ordinary skill in the art to analyze the invention merits in terms of its *metes and bounds*. Figure 2 in light of the Specifications (pg. 6) cannot clarify as to which of the identified entities (i) to (iv) being described therein clearly belong to a given layer; nor do the layers shown in Figure 3 fully mention about 'repository framework model' or 'application object repository framework'; nor do Figure 3b-c elucidate on how the layers exactly correlate the above so-recited entities in the context of *defining*, *generating*, etc. The (first, second) 'layer' limitation will be treated as having applicability being distinct based on the extent to which the claim is understood (e.g. based on data transformation) and interpreted, i.e. using

broad interpretation and what can be gathered from the Specifications without undue strain from the Examiner's standpoint.

Claims 2, 4-5, 22 are also rejected for not remedying to the above.

Claim 18 recites "application framework in the second layer" (li. 4), then "application framework metadata ... occupying a first layer" (li. 8); then 'a repository framework model in the second layer' (li. 13); then 'application object repository framework in the first layer' (li. 12); and along with claims 10, 23 are rejected for not enabling one of ordinary skill in the art to make use of the invention, absent any better clarification from the Specifications.

Claim 19 recites the same inconsistent language as claim 18, and is rejected for indefinite teaching along with claim 12.

Claim 20 and 21 also exhibit the above indefinite nomenclature which cannot be construed based on any helpful clarification from the Specifications; and along with claims 15, 24 are also rejected.

Claims 2, 4-5, 10, 12, 15, 22-24 are also rejected for not remedying to the above.

4. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

5. Claims 2, 4-5, 10, 12, 15, 17-24 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

Claim 17 recites 'defining an application framework in a second layer ... application framework metadata ... occupying a first layer ... defining ... modeling ... application object repository framework in the first layer ... defined by a repository framework model in the second layer'; 'upon validating ... transforming ... into repository framework metadata ... occupying the first layer'. Throughout the Specifications, there is mention of meta-data levels (see Fig. 3), such that each model is comprised of metadata (see Specs pg.7, 2nd para) and levels of meta-data (Specs, pg. 7, bottom half) such that M3 is for meta-meta-model, modeling language for application framework metadata (AFM) is M2, as well as for modeling repository framework metadata (RFM); thus, this is teaching level 2 language is for modeling AFM and RFM. Further, the Specifications discloses that M1 is for AFM simultaneously with repository object model (ROM). All of which is not commensurate with the defining step, generating step and validating step in the exact layer-specific context as claimed respective to the above language, i.e. language being silent about meta-level modeling language for AFM/RFM on one hand, and for AFM/ROM on the other, as disclosed. As set forth in the USC § 112, 2nd para rejection, there is sufficient parallel teaching between the knowledge from reading the Disclosure, the description of Figures and the exact layer-related steps of defining the very entities as identified as (i) to (iv) in the claim. That is, the Specifications describe M-level format of metadata (see above), and layers (e.g. Figure 9, about template workbench in a Rose meta-model, validating templates and generating object repository of source file or scripts to yield a object repository runtime) but nowhere is showing (specific) 'layer' in correlation with the entities **exactly as** phrased in the claim, let alone the confusion created by the M-level meta-format (e.g. Figure 8) paradigm which

also fails to depict layer-pertinent actions (e.g. *defining, generating, modeling* in relation to entities i→iv) as claimed.

What appears to be insufficiency in terms of indefinite language with lack of enablement by the Disclosure (USC 112, 2nd paragraph) is now compounded with layer-related lack of description for all the entities such as (a) *application framework*, (b) *repository framework model* (both in 2nd layer); (c) *application framework metadata*, (d) *application object repository framework* (both in 1st layer). Because the *Specifications* do not show clear sequences involving the above entities in the sense that they are correlated such that one reading the claim can be reasonably apprised that the Inventor possesses the steps as recited, and that the Inventor has provided solid corroborating facts in the Specs with regard to the step actions involving those entities as these are located. A quick keyword search in the Specifications for each of those very entities -- names like (a) to (d) -- will not lead the reader to a single clear and substantially complete portion enabling him/her to perceive that the claimed steps have unequivocal support therefor as far as the first or second layer settings – i.e. implicating terminology (a to d) being concurrently staged with the steps of defining, modeling, transforming, generating. The weight of the *layer* limitation will not be given proper weight and the terminology a-d will be addressed to the extent of the learning (consistent with a very specific context interpretation from the claim) based on the Specifications if and only if when there is sufficient understanding therein.

Claims 18-21 recite the above steps with the layer-related terms identified in (a)-(d) from above; and per analogy with claim 17 are construed as not described with proper support in the Specifications; thus, not enabled in order to help one to make use of the invention.

Claims 2, 4-5, 10, 12, 15, 22-24 for failing to remedy to the above are also rejected. Correction is required.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 2, 4-5, 10, 12, 15, and 17-24 rejected under 35 U.S.C. 103(a) as being unpatentable over Iyengar, USPN: 6,874,146 (hereinafter Iyengar).

As per claim 17, Iyengar discloses a method for generating a software development repository to reflect extensions in an application framework that supports an application, the method comprising:

in a multi-layer modeling architecture, defining an application framework in a second layer using a common modeling language in a third layer (e.g. MOF 20, Fig. 3), wherein the application framework supports an application by providing application constructs and semantics to structure and provide functionality for the application (e.g. *metamodel UML 21* Repository – Fig. 1 – Note: language describing UML represents semantics for Application framework layer using and UML semantics reads on second layer defining of a application framework - see Fig. 4), and wherein application framework metadata representing application framework extensions and occupying a first layer (e.g. Fig. 3; col. 9, line 40 to col. 10, line 19; *extending this capability* - SUMMARY col. 3 - Note: runtime framework to address request coming as first layer application object being modeled via ORB –see *request* -Fig. 1 – reads on first layer constructs

being modeled with runtime framework based on a request, i.e. the application request being in a first layer ORB metamodel) are defined by the application framework, the application framework extensions providing additional application functionality;

defining an application object repository framework related to the application framework by modeling the application object repository framework in the first layer using repository constructs and semantics (e.g. M1 model – Fig. 4) defined by a repository framework model in the second layer (M2 – Fig. 4), wherein the repository framework model is defined by the common modeling language in a third layer (M3 – Fig. 4) that also models the application framework in the second layer (see col. 9, lines 37-54; see Iyengar: Fig. 1, 3B-3C as per USPN: 6038393 - incorporated by reference -- col. 8, lines 8);

responsive to new application framework extensions, generating an application object repository supported by the application object repository framework and conforming to the application framework with the new application framework extensions, comprising:

generating application framework metadata representing the application framework with the new application framework extensions and (*metamodel* UML 21 - Fig. 2; UML- Fig. 4) occupying the first layer (e.g. Fig. 3) in the multi-layer modeling architecture as meta-model data, wherein the application framework metadata is generated using the repository constructs defined by the repository framework model (Fig. 3; Iyengar: Fig. 1, 3B-3C as per USPN: 6038393 - incorporated by reference) in the second layer;

upon said generating, validating the generated meta-model data with respect to the repository framework model constructs (e.g. col. 9, line 40 to col. 10, line 19; *mapping rules* - col. 10, line 41 to col. 42);

upon said validating, transforming the meta-model data into repository framework metadata, the repository framework metadata representing an intermediate representation (e.g. *XML ...produced* - col. 9, lines 24-32) of the application object repository conforming to the application and occupying the first layer of the modeling architecture;

wherein the repository framework metadata includes repository schema metadata and repository runtime metadata (e.g. runtime framework, UML meta-model, Corba DTD, XML streams, Corba objects - col. 9, lines 21-59; Iyengar: Fig. 1, 3B-3C as per USPN: 6038393 - incorporated by reference), wherein the repository schema metadata defines extensions to the schema of the application object repository framework and the repository runtime metadata defines extensions to runtime services (e.g. *extensible object-oriented database application* – col. 7, lines 61-65; see col. 8: *incorporated by reference* col. 8 - USPN: 5,644,764, Johnson et al: e.g. *service 22a, 22b, 22c, 22d*– Fig. 1; col. 3, line 3 to col. 4, line 24) of the application object repository framework;

transforming the repository framework metadata into application object repository source files the source files (e.g. *extensible object-oriented database application* – col. 7, lines 61-65; see col. 8: *incorporated by reference* col. 8 - USPN: 5,644,764, Johnson et al: e.g. *service 22a, 22b, 22c, 22d*– Fig. 1; col. 3, line 3 to col. 4, line 24).

But Iyengar does not explicitly disclose application object repository source file using a predefined transformation template, and a database schema script. However, Iyengar's suggested use of XSL (col. 3, lines 45-52) for rendering browser or a HTTP script -- the XSL being superior constructing tool to build application including XML schema, entails a script being based upon style sheet template in support markup HTTP-based DB query (see Fig. 1, 3, 5;

database collaboration – col. 5, lines 42-65; see Johnson et al, USPN: 5,644,764, incorporated by reference) or HTTP script formatted as XML-based template (see Iyengar: col. 6 line 65 to col. 7 line 40). In view of the teaching from USPN: 5,644,764 regarding services to persist OO data in what appears to be an extensible database as by Iyengar (see col. 7, lines 61-65), one of ordinary skill in the art would utilize this superior use XSL template approach to render a HTTP request using DB query templates expressed by XML specifications thus endeavored by Iyengar, e.g. to implement by means of script extension, thus supporting the remote invocation to databases so instrumental to model-mapping and creating of development objects (in view of the teachings by Johnson --incorporated by reference –also contemplated by Iyengar). The motivation is that by using DB query template inside a HTTP remote call using style-sheet technology (as praised by Iyengar), the HTTP request can be more friendly to developers, and all the while, enables the developer with extensible and easy-to-use means as to retrieve via XML specifications, the data needed for development, taking full advantage of persisted legacy of DB objects and offering extensibility from a imported database resources, e.g. via Object-Oriented tracking by type, attribute, methods and property as by Johnson, such that the OO constructs can be represented as schema of tagged objects as contemplated by Iyengar's XML format (see col. 6, line 52 to col. 7, line 50)

Iyengar does not specifically teach generating an application object repository schema from the database schema script, the application object repository schema defining a relational database structure for storing application metadata representing the application framework extensions. However, based on the XSL approach and a scripting to provide services supporting a OO extension of database and reusing of OO components via such script language as addressed

above, with a construct of a XML representing a DB form of schema (see col. 6, line 52 to col. 7, line 50), the SQL script for supporting relational database structure by Johnson or Iyengar's DB extension would have been obvious for the same reasons as set forth above.

Iyengar does not explicitly disclose compiling the runtime source file to generate an executable component, the executable component providing at least one database service for object-oriented interaction with the stored application metadata in the application object repository.

However, based on the rationale to provide script and OO constructs to support extension of database as taught above (Iyengar: col. 7, line 60 to col. 8, line 45; *developers of ...repositories* , non-Corba ...applications,... development of Corba-based software, database interoperability - col. 9, lines 21-67; - *incorporated by reference*, col. 8; i.e. USPN: 6,018,627 -- hereinafter Iyengar2: *build components, assemble, deploy* –Fig. 2B), one of ordinary skill in the art would be motivated to implement runtime source file supporting this object repository schema and relational structure storage compliant with the services as contemplated by Johnson and extension by Iyengar (see above) such that these source files would be compiled into executable or deployable form as by Iyengar2 because the main purpose of having metadata repository by Iyengar is to support database OO application, interoperability between database (see col. 9, lines 21-67), to extend its functionality (col. 7, lines 61-65) via persistence (see Johnson) and in providing OO metadata to support applications as set forth above, the concept of providing of runtime executable (see Iyengar2) would have been obvious because this necessary means would make use of runtime components assembled via the XML script construct or

schema deploying the services as set forth above (see Iyengar: col. 7, line 60 to col. 8, line 45; USPN: 5,644,764: Fig. 1-7; USPN: 6,018,627: Figs 1-3).

Nor does Iyengar explicitly disclose upon generation of the application object repository, migrating previously generated metadata conforming to earlier application object repositories to the generated application object repository. In light of Iyengar's syntax interchange capability in using XMI and its applicability onto multiple business models and database collaboration or format interchange (e.g. *one repository to another... exchange metadata with other repositories* - col. 10, lines 2-24; Iyengar: *transfer syntax for information interchange, integration ... collaboration ... of database schema information* - col. 5, lines 45-65), it would have been obvious for one skill in the art at the time the invention was made to implement the XMI, the UML repository by Iyengar mapping tool, so that model-based services objects being validated using metadata (see Iyengar: Fig. 5-6) can also be mapped and reconverted according to a migration of repository format via using XMI, i.e. runtime object repository applicable to migrate data supported by one previously stored XML metadata (in one repository) to another metadata and syntax supported by another repository format, in view of the need to interchange syntax due to database format differential as set forth above; thereby enabling Iyengar to accommodate for difference in syntax in exploiting the full benefit of XML (W3c) and its XMI to interchange syntax for the proper business applications using collaboration of persisted XML – as set forth above.

As per claim 2, Iyengar discloses wherein the repository framework metadata is XML ("Extensible Markup Language"): see Fig. 2-4.

As per claims 4-5, Iyengar does not explicitly disclose wherein the at least one service includes versioning, object-oriented access, persistence and change management. But based on the extensibility of Iyengar's database (see col 7, lines 61-65) and the services by Johnson (incorporated by reference USPN: 5,644,764: Fig. 1-7), the DB versioning, access, persistence and change management services would have been obvious services according to the rationale as set forth in claim 17.

Nor does Iyengar explicitly disclose wherein said transforming the meta-model data application into repository framework metadata is achieved using XSL ("Extensible Style Language"). However, the use of scripting to make use of the extensible XML metadata format has been suggested in Iyengar, and rendered obvious in view of the template-based scripting limitation as set forth in claim 17.

As per claim 18, Iyengar discloses a method for generating a software development repository to reflect changes in an application framework that supports an application, the method comprising:

in a multi-layer modeling architecture, defining an application framework in a second layer using a common modeling language in a third layer, wherein the application framework supports an application by providing application constructs and semantics to structure and provide functionality for the application, and wherein application framework metadata representing application framework extensions and occupying a first layer are defined by the application framework, the application framework extensions providing additional application functionality (refer to claim 17);

defining an application object repository framework related to the application framework by modeling the application object repository framework in the first layer using repository constructs and semantics (e.g. M1 model – Fig. 4) defined by a repository framework model in the second layer (M2 – Fig. 4), wherein the repository framework model is defined by the common modeling language in a third layer (M3 – Fig. 4) that also models the application framework in the second layer (see col. 9, lines 37-54; see Iyengar: Fig. 1, 3B-3C as per USPN: 6038393 - incorporated by reference -- col. 8, lines 8);

responsive to new application framework extensions, generating an application object repository supported by the application object repository framework and conforming to the application framework with the new application framework extensions, comprising:

receiving a UML representation (e.g. *metamodel* UML 21 - Fig. 2; UML- Fig. 4) of application framework metadata representing the application framework with the new application framework extensions, and occupying the first layer (e.g. Fig. 3; *extending this capability* - SUMMARY col. 3 - Note: runtime framework to address request coming as first layer application object being modeled via ORB –see *request* -Fig. 1 – reads on first layer constructs being modeled with runtime framework based on a request) in the multi-layer modeling architecture, the application framework metadata specified utilizing predefined UML constructs and repository constructs defined by the repository framework model in the second layer (e.g. Fig. 3; Iyengar: Fig. 1, 3B-3C as per USPN: 6038393 - incorporated by reference; col. 9, line 40 to col. 10, line 19);

upon said receiving, transforming the application framework metadata into XML repository framework metadata representing an intermediate representation (e.g. e.g. *XML*

...*produced* - col. 9, lines 24-32; Fig. 6 – Note: mapping tool using XMI and derived DTD reads on intermediate representation) of the application object repository conforming to the application framework with the new application framework extensions and occupying the first layer of the modeling architecture (*extending this capability* - SUMMARY col. 3 - Note: runtime framework to address request coming as first layer application object being modeled via ORB –see *request* - Fig. 1 – reads on first layer constructs being modeled with runtime framework based on a request) , the repository framework metadata being a function of the repository framework model and the common modeling language (e.g. *XML ...produced* - col. 9, lines 24-32; *mapping rules* - col. 10, line 41 to col. 42),

wherein the repository framework metadata includes repository schema metadata and repository runtime metadata, wherein the repository schema metadata defines extensions to the schema of the application object repository framework and the repository runtime metadata defines extensions to runtime services of the application object repository framework (refer to claim 17);

transforming XML repository framework metadata into application object repository source files including runtime source file (e.g. *extensible object-oriented database application* – col. 7, lines 61-65; see col. 8: *incorporated by reference* col. 8 - USPN: 5,644,764, Johnson et al: e.g. *service 22a, 22b, 22c, 22d*– Fig. 1; col. 3, line 3 to col. 4, line 24).

Iyengar does not explicitly disclose (refer to: transforming the XML repository ... into application object repository source files) using a predefined XSL transformation template, the source file including a database schema script.

However, Iyengar's suggested use of XSL (col. 3, lines 45-52) serving as superior constructing tool to build application from a XML schema entails a script being based upon a XSL template. In view of the teaching from USPN: 5,644,764 regarding services to persist OO data in what appears to be an extensible database as by Iyengar (see col. 7, lines 61-65), one of ordinary skill in the art would utilize this superior use XSL template approach to make use of the XML metadata in order to implement by means of script the extension and development services contemplated by Iyengar in view of the teachings by Johnson (incorporated by reference) to provide persistence of objects and extensibility to a database using Object-Oriented tracking by type, attribute, methods and property as by Johnson, such that the OO constructs can be represented as schema of tagged objects as contemplated by Iyengar's XML format (see col. 6, line 52 to col. 7, line 50);

Iyengar further discloses the application object repository schema defining a relational database structure for storing application metadata representing application development objects and the relations between the application development objects compliant with the application framework changes (e.g. Fig. 3-6 – Note: instance by which UML constructs and MOF definition are instantiated via schema using XML/XMI reads on relations between objects compliant with framework represented by object repository schema defining DB structure – see e.g. Fig. 3; col. 9, line 40 to col. 10, line 19).

But Iyengar does not specifically disclose generating an application object repository schema from the database schema script using OSQL ("object-oriented SQL"). However, based on the XSL approach and a scripting to provide services supporting a OO extension of database and reusing of OO components via such script language as addressed above, with a construct of a

XML representing a DB form of schema (see col. 6, line 52 to col. 7, line 50), the SQL script for supporting relational database structure by Johnson or Iyengar's DB extension would have been obvious for the same reasons as set forth above.

Iyengar does not explicitly disclose compiling the runtime source file to generate an executable component, the executable component providing at least one database service for object-oriented interaction with the stored application metadata in the application object repository; but based on the rationale to provide script and OO constructs to support extension of database as taught above (Iyengar: col. 7, line 60 to col. 8, line 45; *developers of ...repositories* , non-Corba ...applications,... development of Corba-based software, database interoperability - col. 9, lines 21-67; - *incorporated by reference*, col. 8; i.e. USPN: 6,018,627 --hereinafter Iyengar2: *build components, assemble, deploy* –Fig. 2B), one of ordinary skill in the art would be motivated to implement runtime source file supporting this object repository schema and relational structure storage compliant with the services as contemplated by Johnson and extension by Iyengar (see above) such that these source files would be compiled into executable or deployable form as by Iyengar2 because the main purpose of having metadata repository by Iyengar is to support database OO application, interoperability between database (see col. 9, lines 21-67), to extend its functionality (col. 7, lines 61-65) via persistence (see Johnson) and in providing OO metadata to support applications as set forth above, the concept of providing of runtime executable (see Iyengar2) would have been obvious because this necessary means would make use of runtime components assembled via the XML script construct or schema deploying the services as set forth above (see Iyengar: col. 7, line 60 to col. 8, line 45; USPN: 5,644,764: Fig. 1-7; USPN: 6,018,627: Figs 1-3)

As per claim 10, Iyengar does not explicitly disclose wherein the at least one service includes versioning, object-oriented access, persistence and change management. But based on the extensibility of Iyengar's database (see col 7, lines 61-65) and the services by Johnson (incorporated by reference- col. 8- USPN: 5,644,764: Fig. 1-7), the DB versioning, access, persistence and change management services would have been obvious services according to the rationale as set forth in claim 18.

As per claim 19, Iyengar discloses a system for generating an object repository to reflect changes in an application framework, the system comprising:

an interface to receive a visual representation of application framework metadata representing an application framework (e.g. Fig. 3) and changes to the application framework; wherein the application framework metadata conforms to repository constructs defined by an application object repository framework model (e.g. Fig. 3-6 – Note: instance by which UML constructs and MOF definition are instantiated via schema using XML/XMI reads on relations between objects compliant with framework represented by object repository schema defining DB structure – see e.g. Fig. 3; col. 9, line 40 to col. 10, line 19);

a processor; and a memory, coupled to the processor, storing instructions adapted to be executed by the processor to perform:

in a multi-layer modeling architecture, defining an application framework in a second layer using a common modeling language in a third layer, wherein the application framework supports an application by providing application constructs and semantics to structure and provide functionality for the application, and wherein application framework metadata

representing application framework extensions and occupying a first layer are defined by the application framework, the application framework extensions providing additional application functionality (refer to claim 17);

defining an application object repository framework related to the application framework by modeling the application object repository framework in the first layer using repository constructs and semantics (e.g. M1 model – Fig. 4) defined by a repository framework model in the second layer (M2 – Fig. 4), wherein the repository framework model is defined by the common modeling language in a third layer (M3 – Fig. 4) that also models the application framework in the second layer (see col. 9, lines 37-54; see Iyengar: Fig. 1, 3B-3C as per USPN: 6038393 - incorporated by reference -- col. 8, lines 8);

responsive to the application framework changes, generating an application object repository supported by the application object repository framework and conforming to the application framework with the new application framework extensions, comprising:

transform the received application framework metadata into XML repository framework metadata representing an intermediate representation of the application object repository conforming to the application framework with the application framework changes, and occupying the first layer of the modeling architecture, the repository framework metadata being a function of the repository framework model and the common modeling language (refer to claim 18);

transforming XML repository framework metadata into application object repository source files including runtime source file (e.g. *extensible object-oriented database application* –

col. 7, lines 61-65; see col. 8: *incorporated by reference* col. 8 - USPN: 5,644,764, Johnson et al: e.g. *service 22a, 22b, 22c, 22d*– Fig. 1; col. 3, line 3 to col. 4, line 24);

the application object repository schema defining a relational database structure for storing application metadata representing application development objects and the relations between the application development objects compliant with the application framework changes (e.g. Fig. 3-6 – Note: instance by which UML constructs and MOF definition are instantiated via schema using XML/XMI reads on relations between objects compliant with framework represented by object repository schema defining DB structure – see e.g. Fig. 3; col. 9, line 40 to col. 10, line 19).

Iyengar does not explicitly disclose (refer to: transforming the XML repository ... into application object repository source files) using a predefined XSL transformation template, the source file including a database schema script. However, this template and schema script limitations have been addressed in claim 18.

But Iyengar does not specifically disclose generating an application object repository schema from the database schema script using OSQL ("object-oriented SQL"). Nor does Iyengar explicitly disclose compiling the runtime source file to generate an executable component, the executable component providing at least one database service for object-oriented interaction with the stored application metadata in the application object repository.

But all of which step limitations having addressed in claim 18 by virtue of the corresponding obviousness rationale set forth therein.

As per claim 12, Iyengar discloses wherein the visual representation of the application framework metadata utilizes at least a subset of UML ("Unified Modeling Language"): see *metamodel* UML 21 - Fig. 2; UML- Fig. 4.

As per claim 20, Iyengar discloses system to generate an object repository to providing generic migration of previously stored data in a software development repository to reflect changes in an application framework, the system comprising: an interface to receive application framework metadata representing an application framework; a processor; and
a memory, coupled to the processor, storing instructions adapted to be executed by the processor (refer to claim 19) to perform the same step limitations as set forth in claim 17, including the obvious teachings as set forth therein.

As per claim 15, Iyengar discloses further comprising a database storing versions of the application object repository to provide migration of data stored in the application object repository (e.g. SUMMARY: *data interchange ... among repositories* -col. 3, li. 61-67; col. 9, lines 55-56; see incorporated by reference, col. 8: USPN: 5,644,764: Fig. 1-7 – Note: versioning and interoperability between repositories using extensible metadata support reads on migration of data via interchange between DB format).

As per claim 21, Iyengar discloses a method for providing generic migration of previously stored data in a software development repository to reflect changes in an application framework, the method comprising the same step limitations as set forth in claim 17, including the obvious teachings as set forth therein.

As per claims 22-24, Iyengar discloses wherein the application object repository source files are C++ files (incorporated by reference, col. 8: USPN: 6,018,627 – Iyengar2: see col. 9, lines 30-64).

Response to Arguments

8. Applicants' arguments with respect to claims 2, 4-5, 10, 15, 17-24 have been considered but are mostly moot because the new grounds of arguments are only based on changes in the claim amendments. Following are the Examiner's additional remarks.

§ 103 Rejection under Iyengar:

(A) Applicants have submitted that the invention is about re-generating a repository while Iyengar does not disclose generation of repositories, in terms of generating metadata of one metamodel into a different metamodel (Appl. Rmrks pg. 14 bottom to pg. 15 middle). The claim language of the independent claims is somewhat non-succinct, imprecise with respect to the Specifications, verbose and/or repleted with incongruous (unsupported nomenclature) phraseologies rendering it hard to construe the extent to which the invention is done or presented with an understandable utility susceptible to be reconstrued by one attempting to make use of the invention as claimed; and this has been set forth at length in the USC 112 Rejection. Based on interpretation of the claim language deficiency as thus rejected, it is deemed that the portions cited in Iyengar corroborate with the understanding that UML/MOF retrieval and generation of intermediate representation data such as DTD, XMI structures to effectuate a instance of XML being used to map to the application metadata source as set forth in the Rejection; and such generating of meta-information in conjunction with the (USC 103) rationale to migrate format

from one repository to another repository does fulfill what Applicants refer to as creating of metadata from one repository to construct another metamodel within another repository.

(B) Applicants have submitted that there is no metadata validation shown in the Office Action (Appl. Rmrks pg. 15 bottom). In an application framework where metadata is retrieved from one repository in order to provide basis for mapping against model-based constructs of a target business flow, the validating of XML against the model construct is integral to any metadata-based modeling; that is, a must-do process otherwise understood as validating of format or syntax based on programmatic syntactic/semantic requirement of a business process domain -- or an otherwise different repository structure syntax. And such mapping/validating is well-acknowledged and visibly perceived according to well-known paradigm of Iyengar (see Fig. 1-2; refer to col. 10, lines 20-25) not to mention about the other references for validating constructs, using UML, which are also *incorporated by reference* (see Iyengar: col. 8), all of which evidenced in the Office Action.

(C) Applicants have submitted that there is nothing in Iyengar's referring to XSL that enable a motivation for one skill in the art to use XSL as database schema script (Appl. Rmrks pg. 16, middle). A browser script using the metadata format of XML is taught by Iyengar; and template in support of such script is provided by user-friendly style sheet technology; such that when Johnson's UML framework implicate necessary DB queries as Iyengar's browser script extended with a UML-based metadata, all of the above pieces of information are deemed instrumental to and in place enabling how one of ordinary skill would be motivated to effectuate DB query in XML metadata format, using templatized syntax coming from HTTP, XML, DB-specific template or constructor, and finally user-friendly style-sheet form. And this is the rationale of

the obvious rejection. Applicants' "What You See Is What You Get"-editor argument is not sufficient to overcome the above rationale.

In view of the numerous USC § 112 non-compliant issues, and based on the best attempt by the Examiner to interpret how the claimed invention reasonably amounts to in light of the Disclosure, the rejection of the pending claims is maintained as effectuated above, given the insufficiency of the arguments as addressed above.

Conclusion

9. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571)272-3756.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Tuan A Vu
Patent Examiner,
Art Unit 2193
January 24, 2008